Titus Lungu
11/13/2017
Udacity AIND

**Foundations of AI, Project 3: Implement a Domain-Independent Planner**
**Research Review**

One type of popular planning technique in the 1970s was *linear planning*, which strung together subplans and subgoals and assumed totally ordered sequences. This was found to be incomplete, because a complete planner must allow for interleaving of actions from multiple subplans within one sequence (*Russell and Norvig*). In order to alleviate this problem and strive for "real-time" performance, (*Korf, 1987*) introduced the notion of serializable subgoals and minimum lookahead search. Minimum lookahead search searches forward from the current state to a specified depth and then applies the heuristics there, at the search frontier, and then makes a move in the direction of the most beneficial child. This causes interleaving of nodes. Once a plausible move is suggested (based on simulating the moves on the machine and not actually executing them in real time), the agent moves towards that location. This is assuming that operators have uniform cost. When they do not, you also use the cost of the path as a heuristic to estimate the remaining cost. Being able to evaluate, accept, and execute plans discretely rather than having to do so all at once is crucial to real-time or pseudo-real-time AIs. Without the ability to run in real-time, many AI problems wouldn't have any value in being solved because the user would have to wait until the end to receive the solution, which is often needed in parts. Also, incorporate state knowledge and updating information wouldn't be possible either.

These advances led to partial-order logic, which allowed for conflict detection (Tate, 1975a) and avoided interference with conditions that had been met (Sussman, 1975). Later, partial-order logic allowed for proof of completeness and intractability as well. Though falling out of favor in the late 1990s, recent work suggests that perhaps using a planning graph to great heuristics would make partial-order logic useful again. RePOP used the open condition heuristic as well as the relax heuristic to increase the speed of planning through increased greediness and other methods (Nguyen and Kambhampati, 2001).

Bonet and Geffner, 2000, helped revive state space models with their Heuristic Search Planner which made state-space practical for non-traditional planning problems thanks to the ability to reduce the search space using heuristics. State-space is a key and central concept in AI and robotics, allowing the definition of states, actions, and a state transition function. They also used mutexes to ensure compatibility of heuristics and fast and accurate solutions to the planning problems. Drew McDermott also played a large role in the resurgence of state-space models and applied planning to web site interaction as well (2001). Seeing web services as "agents", he used estimated regression planners, which evaluate the difficulty of a goal use backwards propagation and then use that analysis to guide a search through the situation space – using a forward search. An interesting side effect of treating web services as agents is the notion of incomplete knowledge. Usually in AI problems, there is knowledge that the agent does not have and may not know that it doesn't have. There is often also noisy, corrupted data which the agent uses. AI agents must be created to be robust to such real-world scenarios. However, with web services, the agent generally knows when there is incomplete knowledge and what that knowledge is. McDermott calls such terms "learnable terms", and the success or failure of the agent hinges on those unknowns being found out. Unlike other AI agents, a web service agent knows exactly when a planning problem can be solved and when it cannot, and can guarantee it's correct solution. This phenomena makes the way the Internet works possible and, though seemingly a small detail, is crucial to our digital world today.